UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/673,587	09/29/2003	Jonathan Appavoo	YOR920030317US1 (16856)	1607

23389 7590 07/13/2007

SCULLY SCOTT MURPHY & PRESSER, PC
400 GARDEN CITY PLAZA
SUITE 300
GARDEN CITY, NY 11530

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
07/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/673,587	APPAVOO ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 September 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *17 October 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *9/9/03*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed 9/09/2003.

Claims 1-24 have been submitted for examination.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 6, 14, 21 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1, 18 of copending Application No. 11/227,761 (hereinafter '761).

Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations. Following are but a few examples as to how the certain claims from the instant invention and from the above copending application are conflicting with each other.

As per instant claim 6, '761 claim 1 also recites dynamically update an operating system (Note: this is equivalent to *while said operating system remains active ... provides continual availability of hardware resources* of claim 6) comprising identifying references (i.e. *reference pointer*) to said first code component and replacing the identified references to said first code

component to said new code component (i.e. *changing a factory reference pointer ... to the new factory object*). '761 claim 1 does not explicitly recite by separating the first code component into objects grouped in table, whereby references to said objects are entered in the table; but in view of the suggestion by '761 that the reference pointer is changed to point to factory object to new factory object (during the *loading of a new factory object*) and removing the old factory object, it would have been obvious for one skilled in the art to provide said runtime factory object in form of runtime table entry as in a table storing reference entries (each being a pointer); so that these table-represented reference pointers would be identifying during the dynamic replacement of reference objects as recited above, in view of well-known approach using reference table to interrelate dynamic code referencing to provide program references resolution support at runtime.

As per instant claims 14, and 21, these claims recite the main limitations of instant claim 1, hence would also have been obvious variations of '761 claim 1 in light of the above analysis.

As per claims 6, 14, and 21, '761 claim 18 also recites updating of an operating system without rebooting, identifying references (i.e. *determining ... old class definition meets ... requirement of a new class definition; structures indicate ... instantiations of the old class definition* -- Note: structure representing definition of old or new class reads on references of first or new code component; i.e. said structure being identified for said determining leading to the hot-swapping) to said first code component and replacing the identified references to said first code component to said new code component (i.e. *hot swapping each ... object instance for its corresponding old object instance*). '761 claim 18 does not explicitly recite by 'separating the

first code component into objects grouped in table, whereby references to said objects are entered in the table'; but in view of the structure to store the definition referring to the *object instance* being swapped from above, this reference *table* (groups of object reference) limitation would have been obvious in light of the rationale as set forth above.

Objections to Specifications and Claims

4. The **Specifications** is objected to because of the following informalities: paragraph [0013] describes hot swapping in a operating system to replace 'first source code component' (see Summary of Invention: pg. 3) with a 'new source code component'. There is no sufficient teachings throughout the Disclosure to support (insufficient enablement as to) how a 'source code' component is replaced in a hot swap by an OS implemented using multi-processor environment and state safe redirecting , live threads, absent any further mention of 'source code' elsewhere. Correction to this source code in the disclosed method of hot swapping is required.

5. Claims 1-24 for reciting 'first source code' and 'new source code' in that OS runtime context are also objected to because the terms used, in light of the Specifications, do not amount to credible or substantial enabling description, particularly in the context of the method being claimed. That is, this language is to be corrected to be commensurate with the Disclosure, wherein code (C++) components are dynamically instantiated into object instances; i.e. no source code being part of this runtime replacement and indirection process to secure a thread save continual execution. This 'source code' will be treated as mere code instances being source to a replacement step whereby more instances of code are derived.

6. Claims 5, 7-9, 13, 15-17, 20, 22-24 recite 'transferring ... quiescent state' and/or 'quiescent state has been transferred'. There is no description in the Specifications that explicitly

conveys transferring of a quiescent state per se. The quiescent state is disclosed as being 'established' or 'achieved', and that subsequent thereto a transfer (*transfer state from the old instance to the new instance*) is done and relates to 'state of a call' or an object -- no quiescent state being transferred. The above language usage for not being commensurate with the written disclosure is bordering on a lack of description informalities; hence is required to be corrected lest a rejection be effectuated. The state being transferred will be treated as a *state* of a call or related to an object.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 4-9, 12-17, 20-24 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Specifically, claims 4, 12, and 18 recite 'wherein said mechanism is divided'. There is not sufficient antecedent basis for this 'said mechanism' from any parts of the respective base claims. This 'wherein said mechanism is divided' is treated as 'wherein' the base claim method steps or means 'include a multiprocessor system'.

Claims 5-9 recite 'the first code component' and/or 'the new code component'. There appears to be insufficient antecedent basis for the above elements; or lack of teachings as to reasonably conveying how a previously recited 'first source code' or 'new source code component' (re claim 1, li. 2-3) suddenly becomes 'first code component' or 'new code component'. For one of ordinary skill in the art of compiling, there is difference between a

precompiled 'source code component' and a runtime replaced 'code component', particularly when the outset is about a active/dynamic operating system continually servicing its applications within a HW resource context, i.e. the OS described without any compilation step of source code being invoked. The first or new component will be treated as a form of code (without need for compilation) during active runtime of a running OS.

Claims 13-17 exhibit either a 'first code component' and/or 'new code component' hence are also rejected for lack of teachings or antecedent basis, as set forth above.

Claims 20-24 also recite 'first code component' and/or 'new code component' hence are likewise rejected.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1 are rejected under 35 U.S.C. 102(b) as being anticipated by Slingwine et al., USPN: 6,219,690 (hereinafter Slingwine)

As per claim 1, Slingwine discloses in a computer system using an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first source code component, a method of replacing said first source code component with a new source code component while said operating system remains active and while said operating system provides continual availability to applications of the hardware resources (see Fig. 2; *kernel running* - col. 10, li. 54-67), the method comprising:

identifying references to said first source code component (e.g. current generation 108 – pg. 3 – Note: CALLBACK processing to make next generation to become current generation – see col. 6, lines 41-50 – whereby associated structured context information for a thread activity – --- col. 9, li. 45-to col 10, li. 17 – along with generation-contained callback constructs – see Fig. 4; *handle table*, *table_ptrs*, Fig. 6-- are replaced **read on** identifying references of first code component); and

replacing the identified references to said first source code with references to said new source code component (next generation 110, Fig. 3; UPDATES: Add to next generation 90 – Fig. 3; Fig. 4).

As per claims 2-3, Slingwine discloses wherein the method is implemented transparently to said applications (e.g. Fig. 5A, 5B, 5C, 5D; *interrrupt* -- col. 18, lines 42-55 – Note: interrupt periodically invoked without need for user input reads on mutual exclusion of threads via callback implementation being transparent to user – see *kernel 36, interrupt* under *user process* – Fig. 2); wherein the method is scalable (e.g. *global generation, possible large number of processes* – see col. 18, lines 42-46; *expanded* – col. 17, lines 41-67; *per-thread where possible ... some other entity where possible* – col. 10, li. 44-47).

As per claim 4, Slingwine discloses method that is divided across a multiprocessor system so that each processor can proceed independently (e.g. per-processor context - Fig. 4; col. 19, li. 1-3).

As per claim 5, Slingwine discloses establishing a quiescent state for the first code component; transferring said quiescent state from the first code component to the new code

component; and after transferring said quiescent state, swapping the first code component with the new code component (e.g. col. 7, lines 50-67; col. 10, lines 7-51).

As per claim 6, Slingwine discloses separating the first code component into objects; and grouping said objects into a table (e.g. Fig. 6; *one bit per thread ... next level ... group of threads and so on* – col. 9, lines 31-44 – Note: thread and associated data structures per threat activity reads on objects being grouped), whereby references to said objects are entered in the table (e.g. *handle table, table_ptrs*, Fig. 6).

As per claim 7, Slingwine discloses wherein the replacing step includes the steps of: establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component; transferring said quiescent state from the first code component to the new code component; and after transferring said quiescent state, swapping the first code component with the new code component (e.g. col. 7, lines 50-67; col. 10, lines 7-5; Fig. 3-4; *zero overhead mutual exclusion* – col. 5, lines 33-56).

As per claim 8, Slingwine discloses wherein the replacing step includes the steps of: establishing a quiescent state for the first code component; transferring the quiescent state from the first code component to the new code component; and after transferring said quiescent state, swapping the first code component with the new code component (refer to claim 7); and further discloses transferring by providing an infrastructure to negotiate a best transfer algorithm (*zero overhead mutual exclusion* – col. 5, lines 33-56 – Note: zero –overhead via quiescence-based updating (remove and replace) of structures by callback reads on best algorithm).

As per claim 9, Slingwine discloses wherein the step of identifying references includes the steps of

separating the first code component into objects, and grouping said objects into a table, whereby references to said objects are entered in the table (refer to claim 6); and

the replacing step includes the steps of establishing a quiescent state for the first code component; transferring the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and after transferring said quiescent state, swapping the first code component with the new code component (refer to claim 8).

As per claim 10, Slingwine discloses a system for swapping source code in a computer system including an operating system, said operating system including at least one source code component and providing continual availability to applications of hardware resources (refer to claim 1), the system comprising:

means for identifying, while said operating system is active and providing continual access to said resources, references to a first source code component of the operating system; and

means for replacing the identified references, while said operating system is active and providing continual access to said resources (*interrrupt* -- col. 18, lines 42-55 ; *kernel 36, interrupt* under *user process* – Fig. 2), to said first source code with references to a new source code component for the operating system;

all of which steps of identifying and replacing having been addressed in claim 1.

As per claim 11, refer to the rejection of claim 2-3.

As per claims 12-17, refer to claims 4-9, respectively.

As per claim 1, Slingwine discloses a program storage device, for use with a computer system including an operating system to provide access to hardware resources, wherein said

operating system provides access to said resources via a first source code component (Fig. 1-2), said program storage device being readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for replacing said first source code component with a new source code component while said operating system remains active and while said operating system provides continual availability to applications of said resources (refer to claim 1), the method steps comprising:

identifying references to said first source code component; and

replacing the identified references to said first source code with references to said new source code component;

all of which being addressed in claim 1.

As per claims 19-24, refer to claims 4-9, respectively.

Conclusion

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
July 9, 2007